Datamatikkursus

sommer 1968

Specialforelæsning 7.8.68

Jørn Jensen:

Puslespilslægning på GIER.

Programmet, - der bare er skrevet for sjov - kan lægge et simpelt puslespil defineret ved:

Et brædt, bestående af et antal felter i et kvadratnet og

Et sæt af brikker, hver bestående af et antal sammenhængende felter.

Strategi.

Ved kombinatoriske problemer står man som regel over for en søgning i et træ med et overvældende antal grene. For at komme igennem på rimelig tid må man:

1. Forsøge at finde en strategi, der kan kappe dele af træet fra så tidligt som muligt.
2. Vælge sin datarepræsentation på en sådan måde, at søgningen går hurtigt.

I dette program er hovedvægten lagt på (2), idet søgeprocessen udføres af et stykke maskinkode, der genereres ud fra beskrivelsen af brædtet og brikkerne. Programmet illustrerer derfor først og fremmest, hvor langt man kan nå med en behændig datarepræsentation.

Programmet, der er kodet i Gier Algol 4, falder naturligt i 3 dele:

1. Indlæsning og analyse af en grafisk beskrivelse af brædt og brikker, herunder rotation og spejling af brikkerne. Denne del indeholder et illustrerende exempel på brug af en rekursiv procedure.
2. Generering af maskinkode ud fra brædt- og brikbeskrivelserne.
3. Udførsel af den genererede kode og udskrift af løsninger.

De følgende sider indeholder: Et kørselseksempel og et snapshot af den genererede kode for dette eksempel - særligt interesserede kan låne Algol programmet til kopiering.

JJ

KØRSELS EKSEMPEL.

Input:

```
 run <

  XXXX
  XXXX
  XXXX
  XXXX

  XX   XX
 XX    X
  X    X

    X   XXX
    X   X X

  B
```

Output:

```
 print wanted: y

   X X
 X X
   X

 X X
 X
 X

 X
 X

 X X X
 X   X

   4 Pieces in  15 Variants.
  Piece area:  16
  Board area:  16 inside  4 x  4

 X X X X X X
 X         X
 X         X
 X         X
 X         X
 X X X X X X


solution  1

 X X X X X X
 X         X
 X         X
 X         X
 X         X
 X X X X X X


  1 solutions
```

image

EXEMPEL PAA GENEREREET KODE, SNAPSHOT.

Brædtet:

```
522. bs   -1          ,hs   s1  Belagt (1. brik)
523. bs   -1          ,hs   s1  Belagt (1. brik)
524. bs   -1          ,hs   s1  Belagt (2. brik)
525. bs   -1          ,hs   s1  Belagt (2. brik)
526. qq                 LZ      Rand
527. bs   -1          ,hs   s1  Belagt (1. brik)
528. bs   -1          ,hs   s1  Belagt (2. brik)
529. bs   -1          ,hs   s1  Belagt (2. brik)
530. bs   1           ,hs   s1  Frit, søges belagt
531. qq                 LZ      Rand
532. bs   -1          ,hs   s1  Belagt (1. brik)
533. bs   -1          ,hs   s1  Belagt (1. brik)
534. bs   -1          ,hs   s1  Belagt (2. brik)
535. bs   1           ,hs   s1  Frit
536. qq                 LZ      Rand
537. bs   1           ,hs   s1  Frit
538. bs   1           ,hs   s1  Frit
539. bs   1           ,hs   s1  Frit
540. bs   1           ,hs   s1  Frit
541. qq                 LZ      Rand
542. pp   827         ,hv   124   Udhop (R=0: ingen løsning, R≠0: løsning)
```

Stak af placerede brikker:

```
543. pp   537         ,hv   605   4. brik, ikke placeret
544. pp   530         ,hv   580   3. brik, søges placeret
545. pp   524         ,hv   567   2. brik, er placeret
546. pp   522         ,ps   626   1. brik, er placeret
547. qq  (r1)[548]           t1   Hvis afstakning hertil da ingen løsning
548. pi   11
549. is  (r11)[560]   ,it   s512
550. bs   r509 [35]   ,hv   s
551. hvn  r-9 [542]
```

Belæg eller frigiv felter i brædtet:

```
552. gm  (s-4)
553. gm  (s-3)
554. gm  (s-2)
555. gm  (s-1)
556. gm   p
557. hv   s1          X  LT       Forsøg næste brik efter afstakning
```

Brik fundet:

```
558. it   s           ,pt  (r2)[560]  Placer brik i stakken
559. hs   p1          ,qq             Find næste fri felt
560. gs   r-16 [544]          t-1     Stak næste fri felt
561. pp   s                           Start søgning efter brik
```

Brik beskrivelser og søge kode:

```
562. bs   1           ,hv   r6 [568]  Er placeret (som 2. brik)
563. cm   p1          ,hv   r5 [568]        X X
564. cm   p4          ,hv   r4 [568]      X X
565. cm   p5          ,hv   r3 [568]        X
566. cm   p10         ,hv   r2 [568]
567. ga   r-5 [562]   ,hs   r-15 [552]  Brik passer
```

```
568. bs   -1           ,hv   r33 [601]  Ikke placeret
569. cm   p1           ,hv   r8 [577]          X X
570. cm   p5           ,hv   r4 [574]          X
571. cm   p10          ,hv   r3 [574]          X
572. ga   r-4 [568]    ,hs   r-19 [553]
573. cm   p1           ,hv   r4 [577]          X X X
574. cm   p2           ,hv   r3 [577]            X
575. cm   p7           ,hv   r2 [577]
576. ga   r-8 [568]    ,hs   r-23 [553]
577. cm   p5           ,hv   r8 [585]          X
578. cm   p9           ,hv   r4 [582]          X
579. cm   p10          ,hv   r3 [582]        X X
580. ga   r-12 [568]   ,hs   r-27 [553]
581. cm   p5           ,hv   r4 [585]          X
582. cm   p6           ,hv   r7 [589]          X X X
583. cm   p7           ,hv   r2 [585]
584. ga   r-16 [568]   ,hs   r-31 [553]
585. cm   p1           ,hv   r4 [589]          X X
586. cm   p6           ,hv   r3 [589]            X
587. cm   p11          ,hv   r2 [589]            X
588. ga   r-20 [568]   ,hs   r-35 [553]
589. cm   p3           ,hv   r4 [593]          X
590. cm   p4           ,hv   r3 [593]      X X X
591. cm   p5           ,hv   r10 [601]
592. ga   r-24 [568]   ,hs   r-39 [553]
593. cm   p5           ,hv   r8 [601]          X
594. cm   p10          ,hv   r3 [597]          X
595. cm   p11          ,hv   r2 [597]          X X
596. ga   r-28 [568]   ,hs   r-43 [553]
597. cm   p1           ,hv   r4 [601]          X X X
598. cm   p2           ,hv   r3 [601]          X
599. cm   p5           ,hv   r2 [601]
600. ga   r-32 [568]   ,hs   r-47 [553]

601. bs   -1           ,hv   r5 [606]   Ikke placeret
602. cm   p5           ,hv   r2 [604]          X
603. ga   r-2 [601]    ,hs   r-48 [555]        X
604. cm   p1           ,hv   r2 [606]
605. ga   r-4 [601]    ,hs   r-50 [555]        X X

606. bs   1            ,hv   r21 [627]  4. variant placeret som 1. brik
607. cm   p1           ,hv   r10 [617]
608. cm   p2           ,hv   r5 [613]          X X X
609. cm   p5           ,hv   r4 [613]          X   X
610. cm   p7           ,hv   r3 [613]
611. ga   r-5 [606]    ,hs   r-59 [552]
612. cm   p1           ,hv   r5 [617]          X X
613. cm   p6           ,hv   r10 [623]           X
614. cm   p10          ,hv   r3 [617]          X X
615. cm   p11          ,hv   r2 [617]
616. ga   r-10 [606]   ,hs   r-64 [552]
617. cm   p2           ,hv   r5 [622]          X   X
618. cm   p5           ,hv   r9 [627]          X X X
619. cm   p6           ,hv   r3 [622]
620. cm   p7           ,hv   r2 [622]
621. ga   r-15 [606]   ,hs   r-69 [552]
622. cm   p1           ,hv   r5 [627]          X X
623. cm   p5           ,hv   r4 [627]          X
624. cm   p10          ,hv   r3 [627]          X X
625. cm   p11          ,hv   r2 [627]
626. ga   r-20 [606]   ,hs   r-74 [552]
```

Ingen af de ledige brikker kan placeres paa det løbende fri felt:

```
627. hv  (r-67)[560]  X      t1   Afstak sidst placerede brik
```

```
algol <

Puzzle, 22.6.68

begin integer array pieces[1:88]; boolean array board[1:12];
integer array row letter[1:12];
integer total size, n piece, max var size, app, i store, store base,
  entry point, return point, solutions, letter, kind, i, j, m board, n board,
  x, y, min x, max x, x diff, stack 0, old letter;
boolean print, printer, lower, board word;

begin boolean array field[0:99], mask, bit [0:38];
integer array descr [0:10];
integer all bits, piece size, i piece, max pieces, piece,
variants, pos, area,next row, max row, i, j, k, n, r1, r2, r3, board area;

boolean procedure test point (i); value i; integer i;
  begin integer p, q; p:= i:39; q:= i rem 39;
  if integer (field[p] ∧ bit[q]) ≠ 0 then
    begin field[p]:= field[p] ∧ mask[q];
    store descr (i);
    for p:= i-1, i+1, i+r1, i-r1 do test point (p);
    test point:= true
    end
  else test point:= false
  end test point;

procedure print piece (piece); value piece; integer piece;
  begin integer pos, n;
  writecr; writechar(60); piece:= piece + 1; pos:=0;
  for n:= piece rem r3 while piece ≠ 0 do
    begin pos:= pos + n; piece:= piece ÷ r3;
    if pos > r2 then begin writecr; pos:= n:= pos - r2 end;
    for n:= n-1 while n > 0 do begin writechar (0); writechar(0) end;
    writechar(0); writechar(23)
    end;
  writecr; writechar(58)
  end print piece;

procedure store descr (val); value val; integer val;
  begin integer i; i:= piece size:= piece size + 1;
  if i > r2 then writetext (‡<
  Piece size!!  Brik stor‡);
  for i:= i-1 while descr[i] > val do descr[i+1]:= descr[i];
  descr[i+1]:= val
  end;
```

```
start:
select(17);
all bits:= integer 10 39m;
r1:= 39; r2:= 10; r3:= r2+1; max row:= 3800; max pieces:= 100; j:= 1;
for i := 38 step -1 until 0 do
  begin bit[i]:= boolean j; mask[i]:= boolean (all bits - j); j:= j+j end;
for i:=0 step 1 until 99 do field[i]:= false;
descr[0]:= area:= i piece:= max var size:= total size:= n piece:= solutions
   := m board:= n board:= board area:= 0;

Ask: writecr; write text(«Output, intet, skrivemask., perf., linieskr. »);
char:= lyn;
if char=57 then i:=19 else if char=18 then i:=19 else if char=39 then
  i:=35 else if char=35 then i:=11 else go to Ask;
print := char ≠ 57 ≡ kbon; printer := char = 35; select(i);

read pieces:  i:= next row:= 0;
for char := lyn while char ≠ 64 do ;
for char:= lyn while char ≠ 49 ∧ char ≠ 50 do
  begin
  if char = 64 then
    begin next row := next row + r1; i:= next row -1;
    if next row ≥ max row then write text («
  Full field»);
    end
  else if char = 23 then
    begin j:= i ; 39; field [j]:= field [j] ∨ bit [i mod 39] end
  else if char ≠ 0 then
    begin if char = 63 ∨ char = 127 ∨ char = 58 ∨ char = 60 ∨
             char = 639 then i:= i-1 end;
  i:= i+1
  end;

piece size:= 0; next row:= i;
i:= j:= 0;  pos:= 40;
for k:= 0 while integer(field[i]) = 0 do i:= i+ 1;
for j:= j+1 while integer(field[i]) ≠ 0 do
  begin board word:= board[j]:= field[i];  field[i]:= false;  i:= i+1;
  m board:= j;  k:= -1;
  for k:= k+1 while ¬board word do
    begin board word:= board word shift 1; n:= k;  end;
  if n < pos then pos:= n
  end;
for j:= 1 step 1 until m board do
  begin board word:= board[j]:= board[j] shift pos;
  k:= -1;
  for k:= k+1 while integer board word ≠ 0 do
    begin
    if integer( board word ∧ bit[k]) ≠ 0 then board area:= board area + 1;
    board word:= board word ∧ mask[k];
    if k > n board then n board:= k
    end
  end;
n board:= n board + 2;
```

```
for i:= i x 39 step 1 until next row do
  begin
  if test point(i) then
    begin n piece:= n piece + 1; i piece := i piece + 2;
    area:= area + piece size; pieces[i piece]:= piece size;
    if max var size < piece size then max var size:= piece size;
    variants:= 0;
    for j:= 0,1,1,1,2,1,1,1 do
      begin
      if j ≠ 0 then
        begin pos:= piece size:= 0; r1:= r2;
        for pos:= pos + piece mod r3 while piece ≠ 0 do
          begin piece:= piece ÷ r3;
          store descr(pos mod r2 x r2 + (if j=2 then pos÷r2 else r2-1-pos÷r2))
          end
        end;

      piece:= x:= max x:= y:= 0;  min x:= piece size - 1;
      for k:= piece size step -1 until 2 do
        begin
        x diff:= descr[k] mod r1 - descr[k-1] mod r1;  x:= x - x diff;
        if x < min x then min x:= x;  if x > max x then max x:= x;
        if x diff ≤ 0 then begin y:= y + 1; x diff:= x diff + r2 end;
        piece:= (piece + x diff) x r3
        end;
      piece:= piece + x - min x;  r1:= 39;  k:= ipiece;
      if max x - min x + 1 ≥ n board ∨ y+1 > m board then go to next;
      for k:= k + 1 while k ≤ i piece + variants do
        if pieces[k] = piece then go to next;
      if k > max pieces then writetext(‡<
        For mange brikker‡);
      pieces[k]:= piece; variants:= variants + 1;
      if variants = 1 ∧ (print = kb on) then print piece(piece);
next: if char ≠ 49 then begin char:= 49;  go to next piece end;
      end for j;
next piece: total size:= total size+ variants x pieces[i piece] + 1;
    pieces[i piece - 1]:= variants; i piece:= i piece + variants;
    piece size:= 0
    end
  end;

writecr; write cr; writeinteger(‡ddd‡, n piece); writetext(‡< stykker i ‡);
writeinteger(‡d‡, i piece -2x n piece); writetext(‡< varianter,
  Brikareal: ‡); writeinteger(‡d‡, area); write cr; writetext(‡< Brettareal: ‡);
write integer(‡d‡, board area); writetext(‡< indenfor ‡);
write integer(‡d‡, m board); writetext(‡< x ‡); writeinteger(‡d‡, n board-1);
writecr
end read board and pieces;
```

```
app:= total size + n piece + max var size + 12;
code app, store base;
3,44
3,44
arm pa1 , ck -10 ; reserve app word in stack
ga  r1           ;   and set address part of store base
qq      , hs c7  ;   to new last used - 1
arm c35 , hv s2  ;
ga  p1  , gr pa2 ;
qq (pa2) t -1    ;
e;
i store:= store base;

begin
integer array descr[0:max var size × 8], distance[0:max var size -2];
integer i piece, piece count, n var, var size, i var, piece, base,
i descr, pos, set board, piece size, end var, i to var, end to var,
i to sq, i old sq, i, i sq; boolean op;

procedure store (addr, count); value addr, count; integer addr, count;
  begin
  code op, i store, addr, count;
  1, 46
  1, 44
  3, 44
  3, 44
  ps (b1) , arm sa1 ;
  t1 -39  , arm pa4 ;
  t1 -10  , arm pa3 ;
  t1 30   X         ;
  arm sa1 , ps (b2) ;
  qq      V     LT  ;
  gm (sa2) V t 1 M  ;
  gm (sa2)  t 1 MA  ;
  e;
  end store;

op:= 11 631 40 656;        comment pp , hv;    store(0, 0);
  return point:= i store;
for i piece:= 1 step 1 until n piece -1 do     store(0, 0);

op:= 11 631 40 630;        comment pp , ps ;   store(0, 0);
op:= 10 60  46;            comment qq(r)   ;   store(1, 1);
  stack 0:= istore;
op:= 10 635;               comment pi      ;   store(0, 0);
op:= 11 636 46 637 41;     comment is(r), its; store(max var size+6, -512);
op:= 11 649 42 656 41;     comment bs r , hvs; store(-515, 0);
op:= 10 656 410;           comment hvn r   ;   store(- npiece - 5, 0);

op:= 10 626 45;            comment gm (s);
for i sq:= -max var size+1 step 1 until -1 do  store(i sq, 0);

op:= 10 626 43;            comment gmp;        store(0, 0);
op:= 10 656 41 34 23 34;   comment hv s X LT;  store(1, 0);
op:= 11 637 41 633 46;     comment it s,pt(r); store(0, 2);
op:= 11 650 43;            comment hs p,;      store(1, 0);
op:= 10 661 42;            comment gs r;       store(-max var size-8, -1);
  entry point:= i store;
op:= 10 631 41;            comment pp s;       store(0, 0);
```

```
i piece:= 1; set board:= -4;
for piece count:= 1 step 1 until n piece do
  begin
  n var:= pieces[i piece]; var size:= pieces[i piece+1]; i piece:= i piece+2;
  set board:= set board - 1; piece size:= n var×var size; idescr:= 0;
  op:= 11 649 40 656 42;  comment bs, hvr;  store(-1, piece size+1);
  for i var:= 1 step 1 until n var do
    begin  piece:= pieces[i piece] + 1; i piece:= i piece + 1;
    base:= piece mod 11; pos:= 0;
    for pos:= pos + piece mod 11 while piece ≠ 0 do
      begin  piece:= piece ÷ 11;
      descr[i descr]:= pos ÷ 10 × n board + pos mod 10 - base; i descr:= i descr +
      end
    end;

  for i var:= 1 step var size until piece size do
    begin end var:= i var + var size -2;
    for i sq:= i var step 1 until end var do
    begin distance[i sq-i var]:= piece size - i sq+1;
      for i to var:= i var + var size step var size until piece size do
        begin end to var:= i to var + var size -2;
        for i to sq:= i to var step 1 until end to var do
          if descr[i sq] = descr[i to sq] then go to end i to var;
        for i to sq:= i to var step 1 until end to var do
          begin
          for i old sq:= i sq-1 step - 1 until i var do
            if descr[i old sq] = descr[i to sq] then go to end i to sq;
          distance[i sq - i var]:= i to sq - i sq; go to end i sq;
          end i to sq:
          end;
        end i to var:
        end;
      end i sq:
      end;
    set board:= set board - var size;
    op:= 11 638 43 656 42;  comment cm p, hv r;
    for i sq:= i var step 1 until end var do
      store(descr[i sq], distance[i sq- i var]);
    op:= 11 622 42 650 42;  comment ga r, hs r;
      store(-end var-1, set board-var size)
    end i var

  end piece count;

op:= 10 656 46 11;  comment hv (r) X;  store(-total size-2, 1);
end store piece code;
```

```
app:= m board × n board;

code app, store base;
3, 44
3, 44
     arn pa1    , ck  -10    ;
     ga  r1                  ;
     qq         , hs  c7     ;
     arn c35    , hv  s2     ;
     ga  p1     , gr  pa2    ;
     qq (pa2)   t   -1       ;
e;

next try:
if print = kb on then
  begin writecr; writecr; i store:= store base;
  if solutions ≠ 0 then
    begin writetext(«<Løsning »); writeinteger(«dd»,solutions); writecr
    end;
  lower:= true;
  for i:= 0 step 1 until m board do
    begin  writecr; writechar(0); old letter:= 0;
    for j:= 1 step 1 until n board do
      begin letter:= 0;
      if i = 0 then row letter[j]:= 0;
      if i < m board then
        begin
        if solutions ≠ 0 then
          code 1 store, letter;
          3, 44
          3, 44
     pm (pa1)  t    1       ;
     tln  9    , gr  pa2    ;
          e
        else letter:= if board[i+1] shift j then 1 else 0
        end;

      if row letter[j] ≠ old letter then
        begin if lower then writechar(60); lower:= false;
          writechar(14); if printer then go to p2
        end;
      write char(0);
p2:   if letter ≠ row letter[j] then
        begin if ¬lower then writechar(58); lower:= true;
           writechar(14); if printer then go to p3
        end;
      writechar(0);
p3:   old letter:= row letter[j]; row letter[j]:= letter
      end for j
    end for i;
  writecr; if ¬lower then writechar(58)
  end print board;
```

```
i store:= store base;
for i:= 1 step 1 until m board do
  begin boardword:= board[i];
  for j:= 1 step 1 until n board do
    begin boardword:= boardword shift 1;
    kind:= if boardword then solutions else -1;
    code i store, kind;
     3, 44
     3, 44
     arn pa2   , pm  re1   ;   i store:= i store + 1;
     pm  re2         LZ    ;   if kind = -1 then
     pm  re3   V     LF    ;     store[i store] M:= qq LZ
     gm (pa1)  Vt  1 MA    ;   else
     gm (pa1)   t  1 M     ;     store[i store] MA:= if kind = 0 then
     hv  re4               ;       bs 1  hs s1  else bs -1  hs s1;
e1:  bs  -1    , hs  s1    ;
e2:  bs  1     , hs  s1    ;
e3:  qq              LZ    ;
e4:                        ;
    £
    end for j
  end for i;

code store base, return point, entry point, solutions, i, stack 0;
     3,44
     3,44
     3,44
     3,44
     3,44
     3,44
     it  re3   , pt (pa2)  ;
     is (pa6)  , pi (si)   ;
     gp (pa2)  , pm  re1   ;
     ps (pa3)  , arn pa4   ;
     arn re2   VX       LZ ;
     arn re2   , hv (s)    ;
     pp (pa1)  , hv  s-1   ;
e1:  bs  1     , hs  s1    ;
e2:  bs  -1    , hs  s1    ;
e3:  gr  pa5   , pa  re5   ;
     hv  re9           LZ  ;
e4:  ps (pa6)  , arn s-1   ;
     ck  10    , ga  re8   ;
e5:  arn [letter] D t 1    ;
e6:  pa  re7   , is (s-1)  ;
e7:  ga s[rel sq], it -1   ;
e8:  arn [piece sq], ga re7;
     ck  20 , ca 611[=cmp] ;
     arn re5   , hh  re6   ;
     ps  s-1   , is (pa3)  ;
     arn(s)    D           ;
     nc  s     , hh  re4   ;
e9:                        ;
£;

if i ≠ 0 then begin solutions:= solutions + 1; go to next try end;
writecr; writecr; writeinteger(‡dd‡,solutions);
writetext(‡< løsninger‡); writecr;
end

run<
```